Dataset :

Super Store Sales analysis:

Load Data Set :

```
df = pd.read_csv("Sample - Superstore.csv", encoding='ISO-8859-1')
df.head()
```

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Postal Code	Region	Product ID	Category	Sub-Category	Produc Nam
0	1	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-BO-10001798	Furniture	Bookcases	Busl Somerse Collectio Bookcas
1	2	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-CH-10000454	Furniture	Chairs	Hon Delux Fabri Upholstere Stackin Chairs,.
2	3	CA-2016-138688	6/12/2016	6/16/2016	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	90036	West	OFF-LA-10000240	Office Supplies	Labels	Self Adhesiv Addres Labels fo Typewriter b.
3	4	US-2015-108966	10/11/2015	10/18/2015	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	FUR-TA-10000577	Furniture	Tables	Bretfor CR450 Series Slin Rectangula

Perform Task :

1. Shape

```
df.shape # how much row and column
```

```
(9994, 21)
```

2. Columns

```
df.columns # Columns name
```

```
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'Country', 'City', 'State',
       'Postal Code', 'Region', 'Product ID', 'Category', 'Sub-Category',
       'Product Name', 'Sales', 'Quantity', 'Discount', 'Profit'],
      dtype='object')
```

3. Information of Data

```
df.info() # table information
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Row ID          9994 non-null   int64
 1   Order ID        9994 non-null   object
 2   Order Date      9994 non-null   object
 3   Ship Date       9994 non-null   object
 4   Ship Mode       9994 non-null   object
 5   Customer ID     9994 non-null   object
 6   Customer Name   9994 non-null   object
 7   Segment         9994 non-null   object
 8   Country         9994 non-null   object
 9   City            9994 non-null   object
 10  State           9994 non-null   object
 11  Postal Code     9994 non-null   int64
 12  Region          9994 non-null   object
 13  Product ID      9994 non-null   object
 14  Category        9994 non-null   object
 15  Sub-Category    9994 non-null   object
 16  Product Name    9994 non-null   object
 17  Sales           9994 non-null   float64
 18  Quantity        9994 non-null   int64
 19  Discount        9994 non-null   float64
 20  Profit          9994 non-null   float64
dtypes: float64(3), int64(3), object(15)
memory usage: 1.6+ MB
```

4. Duplicate value

```
df.duplicated().sum() # Find duplicate in given data

np.int64(0)
```

5. Business Performance

```
# Total sales , profit, orders , unique customers
total_sales = df['Sales'].sum()
total_profits = df['Profit'].sum()
unique_customers = df['Customer Name'].nunique()
total_orders = df['Order ID'].nunique()
df.head(793)

total_sales , total_profits , unique_customers, total_orders

(np.float64(2297200.8603000003), np.float64(286397.0217), 793, 5009)
```

6. Maximum Sales

```
# find maxmimum sales
df.groupby('Category')['Sales'].max()

Category
Furniture          4416.174
Office Supplies    9892.740
Technology        22638.480
Name: Sales, dtype: float64
```

7. Minimum Sales

```
# find minimum sales
df.groupby('Category')['Sales'].min()

Category
Furniture          1.892
Office Supplies    0.444
Technology         0.990
Name: Sales, dtype: float64
```

8. Total sales by Year and Month

```
# aggregates total sales by Year and Month
df['Year'] = df['Order Date'].dt.year
df['Month'] = df['Order Date'].dt.month_name()
df.groupby('Year')['Sales'].sum()
df.groupby('Month')['Sales'].sum().sort_values(ascending = False)

Month
November     352461.0710
December     325293.5035
September    307649.9457
March        205005.4888
October      200322.9847
August       159044.0630
May          155028.8117
June         152718.6793
July         147238.0970
April        137762.1286
January       94924.8356
February      59751.2514
Name: Sales, dtype: float64
```

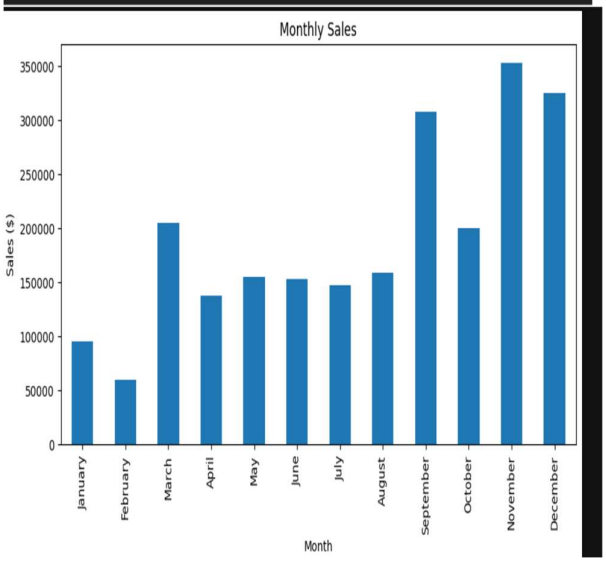
9. Finds the mean profit

```
# finds the mean profit
df.groupby('Region')['Profit'].mean()

Region
Central     17.092709
East        32.135808
South       28.857673
West        33.849032
Name: Profit, dtype: float64
```

10. Monthly sales using plot

```
# monthly sales using plot
monthly_sales = df.groupby('Month')['Sales'].sum()
monthly_sales = monthly_sales.reindex([
    'January','February','March','April','May','June',
    'July','August','September','October','November','December'
])
monthly_sales.plot(kind='bar', figsize=(10,5), title='Monthly Sales')
plt.ylabel('Sales ($)')
plt.show()
```



11. Generate highest profit

```
# generate the highest overall profit
df.groupby('Sub-Category')['Profit'].sum().sort_values(ascending = False)
```

```
Sub-Category
Copiers          55617.8249
Phones           44515.7306
Accessories      41936.6357
Paper            34053.5693
Binders          30221.7633
Chairs           26590.1663
Storage          21278.8264
Appliances       18138.0054
Furnishings      13059.1436
Envelopes         6964.1767
Art               6527.7870
Labels            5546.2540
Machines          3384.7569
Fasteners          949.5182
Supplies         -1189.0995
Bookcases        -3472.5560
Tables          -17725.4811
Name: Profit, dtype: float64
```

11. Generate highest profit

```
# generate the highest overall profit
df.groupby('Sub-Category')['Profit'].sum().sort_values(ascending = False)
```

```
Sub-Category
Copiers        55617.8249
Phones         44515.7306
Accessories    41936.6357
Paper          34053.5693
Binders        30221.7633
Chairs         26590.1663
Storage        21278.8264
Appliances     18138.0054
Furnishings    13059.1436
Envelopes       6964.1767
Art             6527.7870
Labels          5546.2540
Machines        3384.7569
Fasteners        949.5182
Supplies       -1189.0995
Bookcases      -3472.5560
Tables        -17725.4811
Name: Profit, dtype: float64
```

12. use for date like [year – month- day]

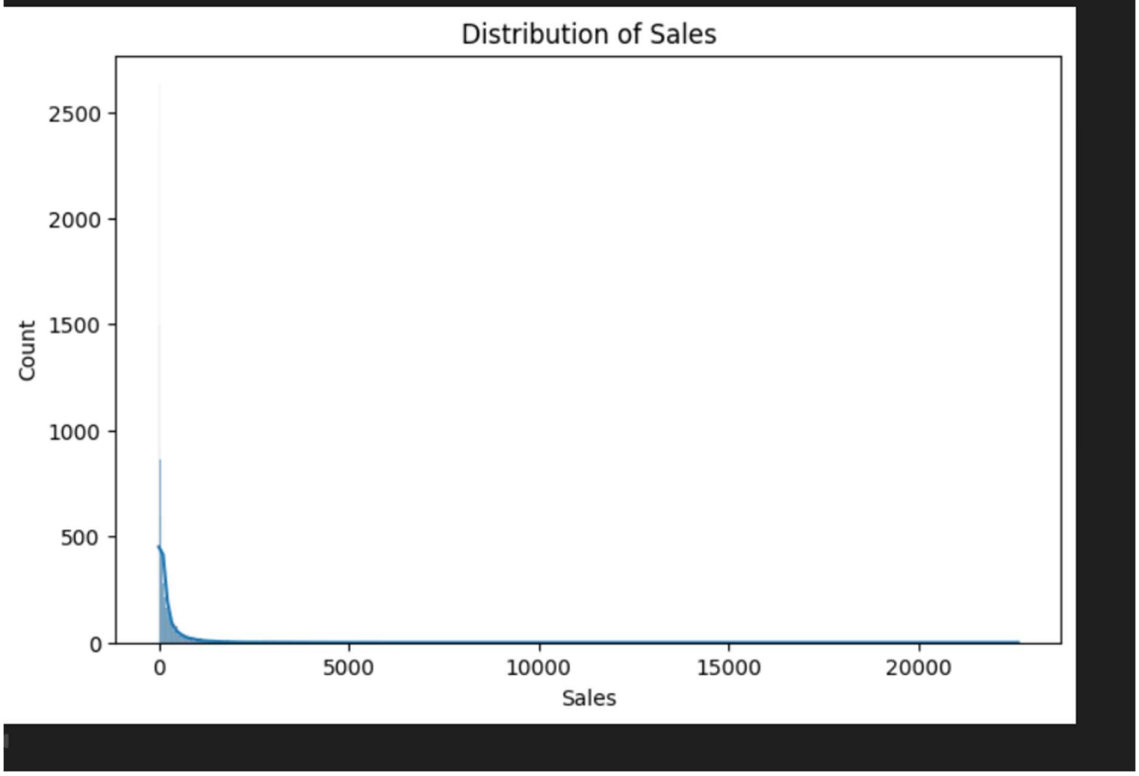
```
# Use for date like [ year-month-day]
df['Order Date'] = pd.to_datetime(df['Order Date'])
df['Ship Date'] = pd.to_datetime(df['Ship Date'])
df.head(2) # Use for date like [ year-month-day]
```

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Postal Code	Region	Product ID	Category
0	1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-BO-10001798	Furniture
1	2	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-CH-10000454	Furniture

2 rows × 21 columns

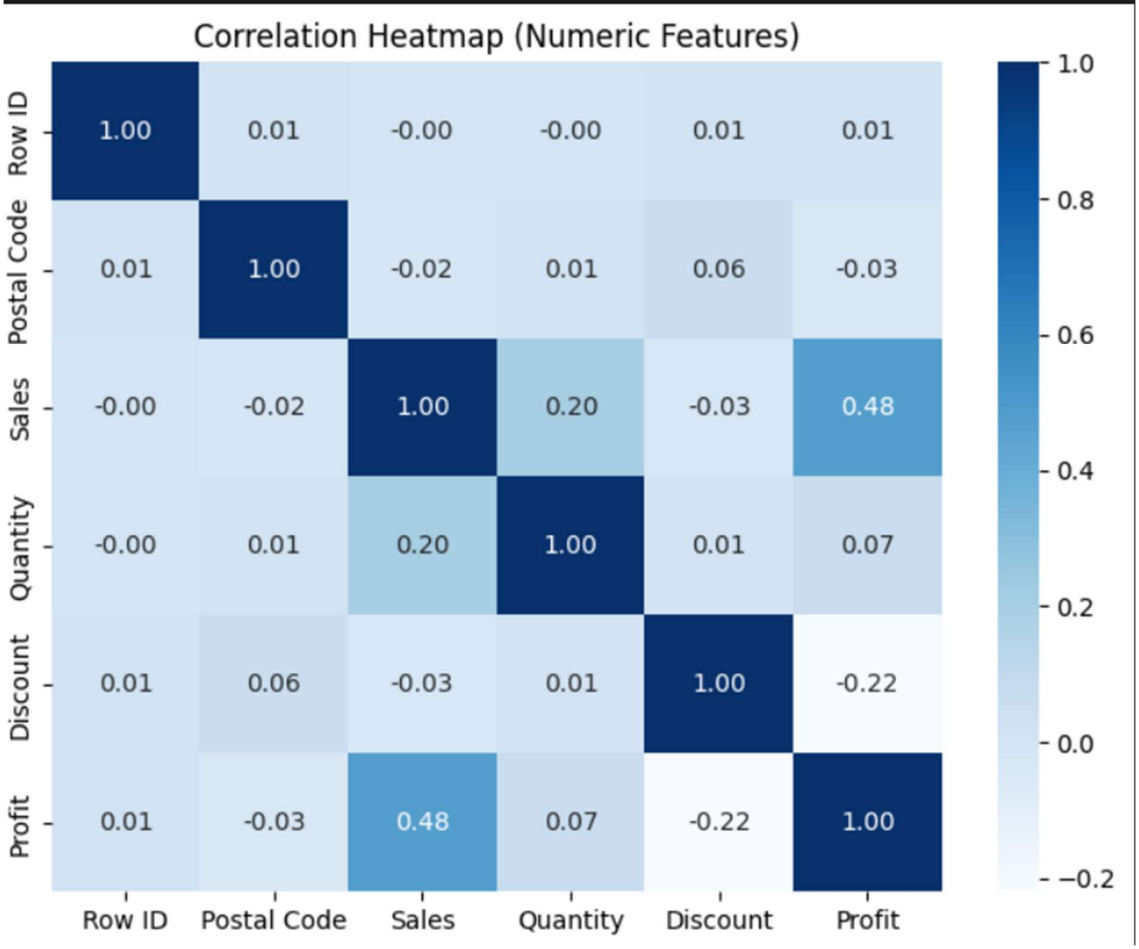
13. Distribution of Sales

```
plt.figure(figsize=(8,5))
sns.histplot(df['Sales'], kde=True)
plt.title("Distribution of Sales")
plt.show()
```



14. Correlation Heatmap(Numeric Features)

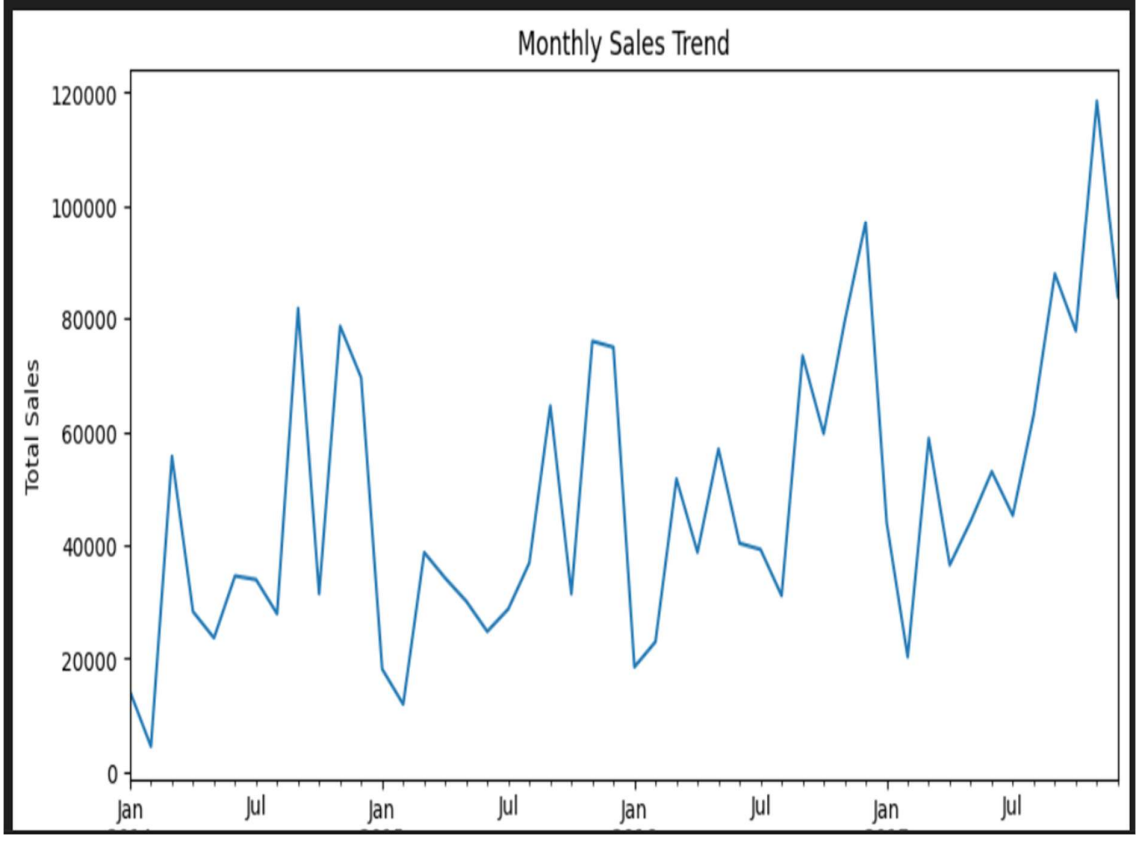
```
plt.figure(figsize=(8,6))
numeric_df = df.select_dtypes(include=['int64', 'float64'])
sns.heatmap(numeric_df.corr(), annot=True, fmt=".2f", cmap="Blues")
plt.title("Correlation Heatmap (Numeric Features)")
plt.show()
```



15. Monthly Sales Trend

```
df_time = df.groupby(df["Order Date"].dt.to_period("M"))["Sales"].sum()

plt.figure(figsize=(10,5))
df_time.plot()
plt.title("Monthly Sales Trend")
plt.xlabel("Year-Month")
plt.ylabel("Total Sales")
plt.show()
```



16. Compare Sales Vs Profit

```
plt.figure(figsize=(7,5))
sns.scatterplot(x="Sales", y="Profit", data=df)
plt.title("Sales vs Profit")
plt.show()
```

17. Total Profit by Segment

```
plt.figure(figsize=(7,5))
sns.barplot(x="Segment", y="Profit", data=df, estimator=sum, palette="viridis")
plt.title("Total Profit by Segment")
plt.show()
```

```
C:\Users\tarpi\AppData\Local\Temp\ipykernel_1964\105888946.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x="Segment", y="Profit", data=df, estimator=sum, palette="viridis")
```